package com.twitter.search.earlybird\_root.filters;

import java.util.Optional;

import javax.inject.Inject;

import com.google.common.base.Preconditions;

import com.google.common.collect.Lists;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.decider.SearchDecider;

import com.twitter.search.earlybird.common.ClientIdUtil;

import com.twitter.search.earlybird.thrift.EarlybirdRequest;

import com.twitter.search.earlybird.thrift.EarlybirdResponse;

import com.twitter.search.earlybird.thrift.EarlybirdResponseCode;

import com.twitter.search.earlybird.thrift.ThriftSearchResult;

import com.twitter.search.earlybird.thrift.ThriftSearchResults;

import com.twitter.search.earlybird\_root.quota.ClientIdQuotaManager;

import com.twitter.search.earlybird\_root.quota.QuotaInfo;

import com.twitter.util.Future;

public class DisableClientByTierFilter extends SimpleFilter<EarlybirdRequest, EarlybirdResponse> {

private static final String CLIENT\_BLOCKED\_RESPONSE\_PATTERN =

"Requests of client %s are blocked due to %s disable";

private final SearchDecider decider;

private final ClientIdQuotaManager quotaManager;

/\*\*

\* Construct the filter by using ClientIdQuotaManager

\*/

@Inject

public DisableClientByTierFilter(ClientIdQuotaManager quotaManager, SearchDecider decider) {

this.quotaManager = Preconditions.checkNotNull(quotaManager);

this.decider = decider;

}

@Override

public Future<EarlybirdResponse> apply(EarlybirdRequest request,

Service<EarlybirdRequest, EarlybirdResponse> service) {

String clientId = ClientIdUtil.getClientIdFromRequest(request);

Optional<QuotaInfo> quotaInfoOptional = quotaManager.getQuotaForClient(clientId);

QuotaInfo quotaInfo = quotaInfoOptional.orElseGet(quotaManager::getCommonPoolQuota);

// Tier value should exist: if client's tier value not in config file, it will be

// set to "no\_tier" by default in ConfigBasedQuotaConfig

String tier = quotaInfo.getClientTier();

Preconditions.checkNotNull(tier);

if (decider.isAvailable("superroot\_unavailable\_for\_" + tier + "\_clients")) {

return Future.value(getClientBlockedResponse(clientId, tier));

} else {

return service.apply(request);

}

}

private static EarlybirdResponse getClientBlockedResponse(String clientId, String tier) {

return new EarlybirdResponse(EarlybirdResponseCode.CLIENT\_BLOCKED\_BY\_TIER\_ERROR, 0)

.setSearchResults(new ThriftSearchResults()

.setResults(Lists.<ThriftSearchResult>newArrayList()))

.setDebugString(String.format(CLIENT\_BLOCKED\_RESPONSE\_PATTERN, clientId, tier));

}

}